おねがい

- 今日の演習はペアで作業します
- 二人でひとつのPCを操作してください
- 相手がつまずいてたら質問してね
  - 早く終わってしまったら...

dRuby

分散Rubyの体験

Masatoshi SEKI / @m_seki

Agenda

- 私について
- dRubyの紹介
- 演習

重要なことを先に

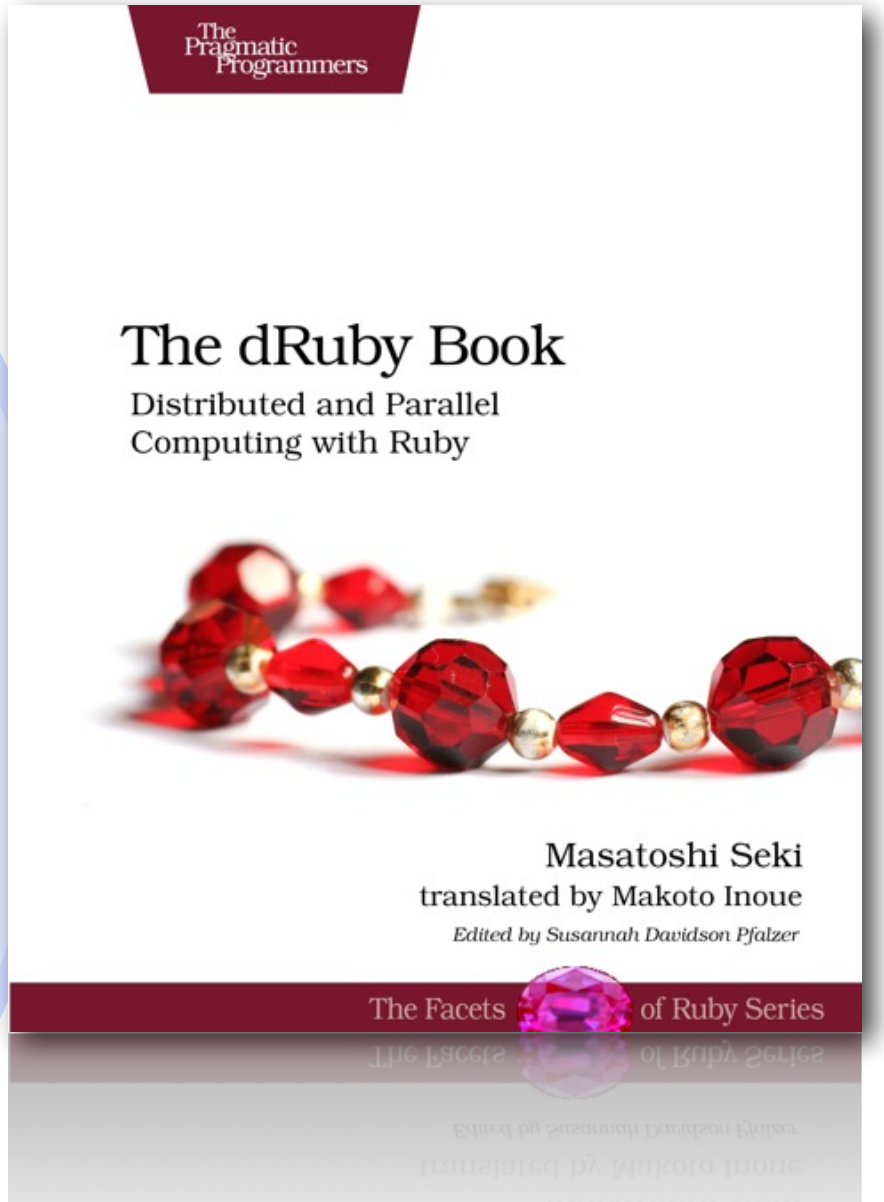
- 先に

重要

- 2005年日本先行発売
- まだ初刷買えます!

The dRuby Book

- 2012年全世界向け
- Out of print



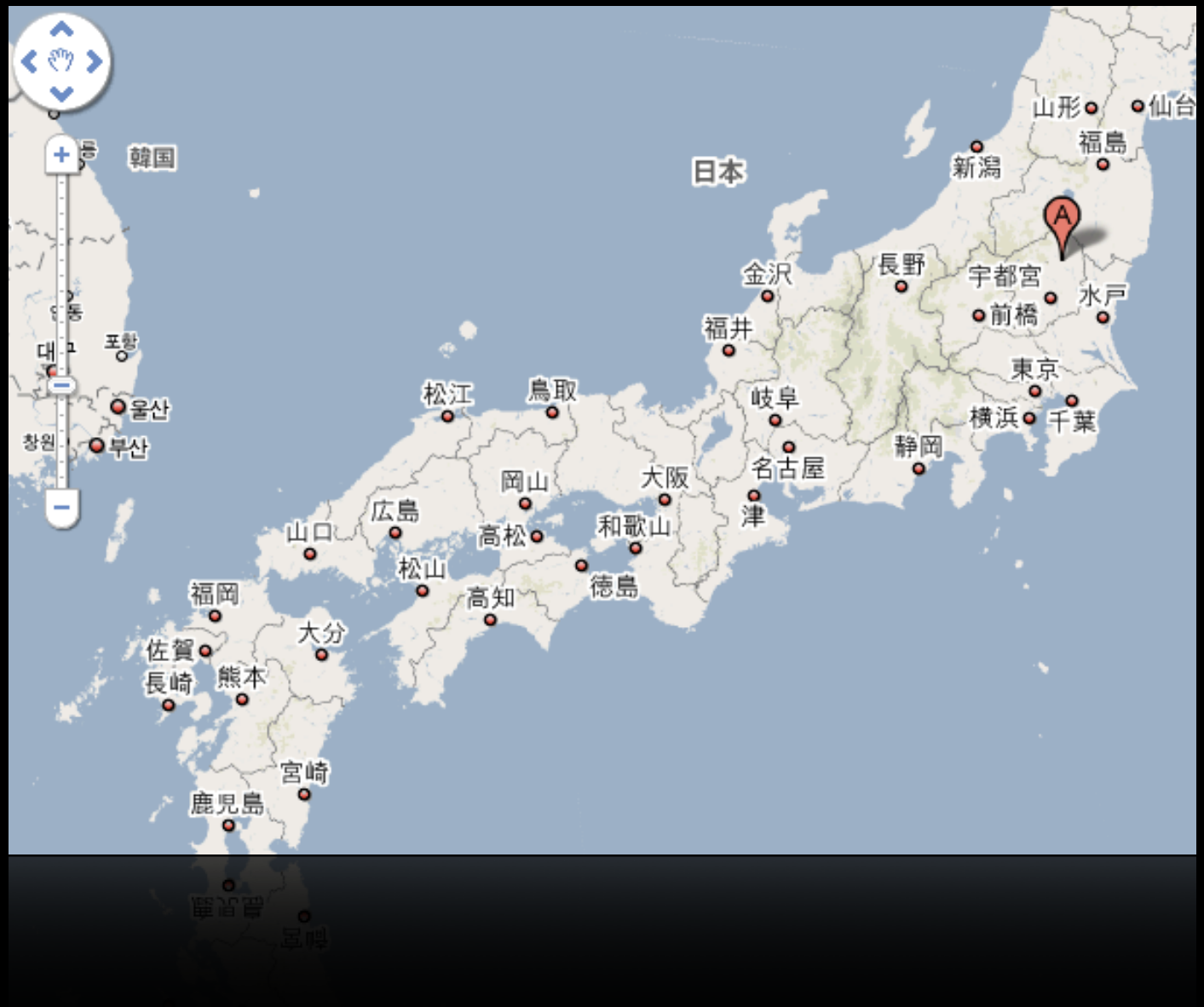
HTML版公開中

- タダ！
  - http://www.druby.org/sidruby/

私について

- 地理的な関係
- しごと関係
- Rubyとの関係

栃木県那須塩原市


韓国
日本
山形
仙台
福島
新潟
A
金沢
長野
宇都宮
水戸
前橋
福井
東京
横浜
千葉
静岡
岐阜
名古屋
津
大阪
松江
鳥取
岡山
広島
山口
高松
和歌山
松山
徳島
高知
福岡
大分
佐賀
長崎
熊本
宮崎
鹿児島
울산
부산
포항

toRuby

- とちぎRubyの勉強会 - 来週147回
  - 毎月第一水曜日 18:30
  - 西那須野公民館
  - クルマ🚚でも15時間で着く

ひさびさの松江！

- 島根大の講義から7年ぶり！？
  - 私は✈️

しごと

- サラリーマン
- 自称アーティスト

自称アーティスト

- 自分のため・自己中心
- アーティストとしてのプログラマ
- わかって欲しい人にウケるため

Rubyと私

- 1999年頃から
- ERB, dRuby, ...
- 2000年のPerl/RubyConference
- 書籍・雑誌

ERB

- 文書にRubyスクリプトを埋め込む
- Railsの実習で触った?
- 非常にこだわりがあるライブラリ

Rubyist Magazine

去年松江に来たちょっと有名な人

Chad Fowler on Ruby

Rubyとの出会い

――いつ、どうやって Ruby と出会ったのでしょうか？

僕はすぐに文法を調べて、ERb (“Embedded Ruby”) を見つけた。そして、 MySQL をバックエンドにした CGI フレームワークとして Ruby を使って、自分のブログシステムを書き直してデプロイした。お昼ごはんを食べる頃にはだいたいできていたかな。このときまでにそれなりの数の「今週の言語」をこなしていたんだけど、こんなに早く進められたことはなかった。どうかしてると思ったよ。日曜日になって、僕はまた Ruby を使った。月曜日の夜、仕事が終わったあとも。次の日も。また次の日も。

Ruby と出会って良くなかったのは、結局 Smalltalk が上達しなかったことだね。

dRuby

- 分散Ruby

今日は

- dRubyの基本的な機能を紹介し、実際に動かして実験します
- 細かいどうでもよい点（しかし私がていねいに設計した点）にはなるべく触れません
- 雰囲気を味わってください

今日は

- 一度にたくさんのプログラムを起動します
  - やったことありますか？

RMI

- Remote Method Invocation
- 別のプロセス／マシンのオブジェクトにメッセージを送り、メソッドを起動する仕組み

越えられない壁

- プロセスの壁
  - OSによって保護された空間
  - 別プロセスのメソッドは呼べない

ふつうのプロセスは

- 聞く耳を持たない
- 自分から周りの声を聴こうとしなくてはわからない
  - 例) 自分からファイルを読み書き とか

サーバ

- 誰かに奉仕するプロセス
- たとえばWebサーバ
  - それが本業であるプログラム
  - そうなるべく最初から設計されてる
  - お願いを待ち受ける仕組を持つ

dRubyは

- みんなをサーバにするライブラリ
- どんなプロセスにも待ち受ける仕組みを提供する

Ruby風

- Webサービスとの違い
  - dRubyはRubyに特化している
  - 徹底的にRubyの常識に従う

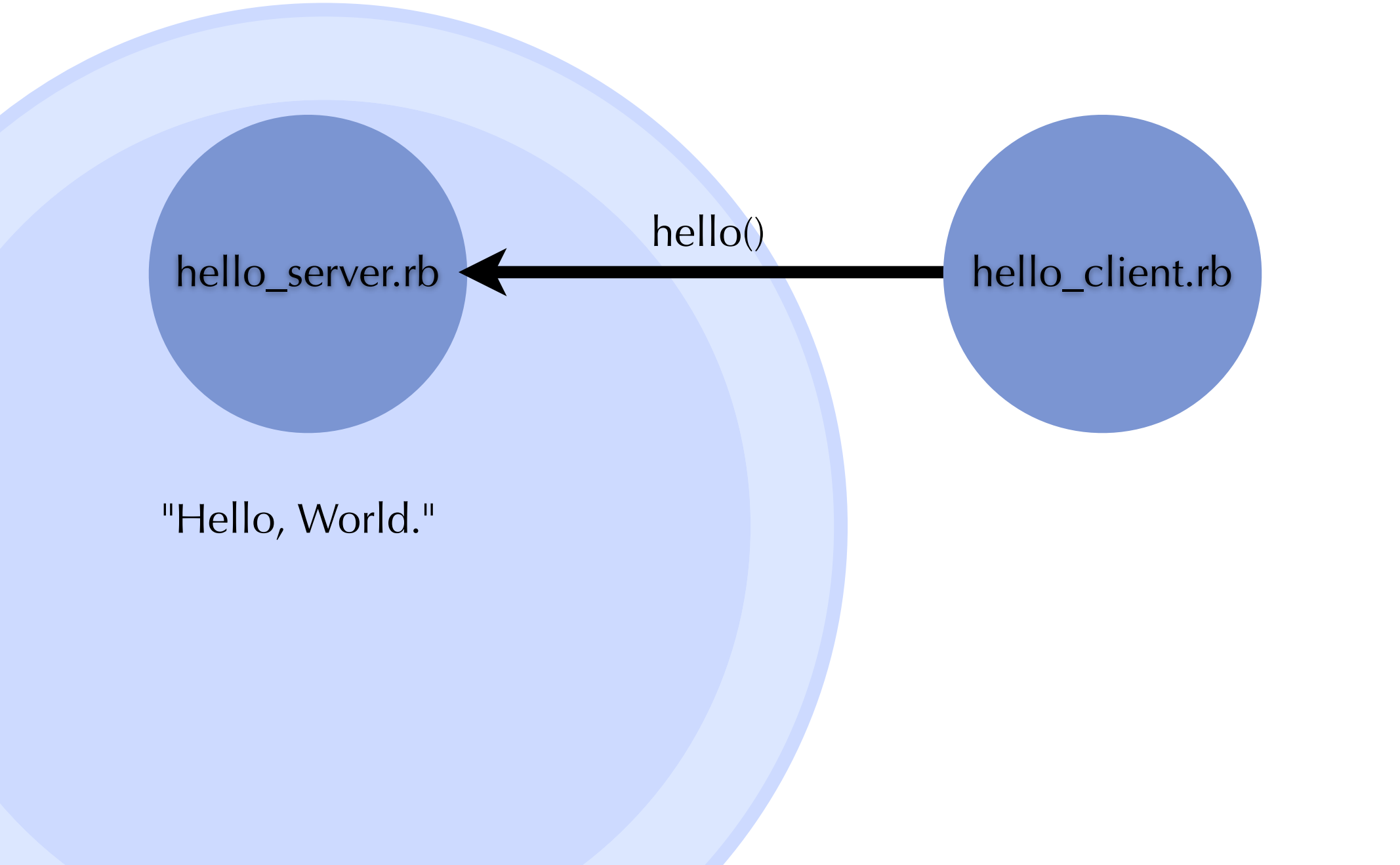
演習

- 1台のマシンの中の複数のプロセスが協調して動く様子を体験します

演習1

- Hello, World.
- いくつかの約束事を覚えます

演習1



hello_server.rb

```
require 'drb/drb'

class Hello
  def hello
    puts('Hello, World.')
  end
end

DRb.start_service('druby://localhost:54000', Hello.new)
while true
  sleep 1
end
```

require

```
require 'drb/drb'

class Hello
  def hello
    puts('Hello, World.')
  end
end

DRb.start_service('druby://localhost:54000', Hello.new)
sleep
```

DRb.start_service

```
require 'drb/drb'

class Hello
  def hello
    puts('Hello, World.')
  end
end

DRb.start_service('druby://localhost:54000', Hello.new)
sleep
```

URI

```
require 'drb/drb'

class Hello
  def hello
    puts('Hello, World.')
  end
end

DRb.start_service('druby://localhost:54000', Hello.new)
sleep
```

終了させない

```
require 'drb/drb'

class Hello
  def hello
    puts('Hello, World.')
  end
end

DRb.start_service('druby://localhost:54000', Hello.new)
sleep
```

約束事

- require 'drb/drb'
- DRb.start_service
- URI
- 終了させない

hello_client.rb

```
require 'drb/drb'

DRb.start_service
ro = DRbObject.new_with_uri('druby://localhost:54000')
ro.hello
```

hello_client.rb

```
require 'drb/drb'

DRb.start_service
ro = DRbObject.new_with_uri('druby://localhost:54000')
ro.hello
```

hello_client.rb

```
require 'drb/drb'

DRb.start_service
ro = DRbObject.new_with_uri('druby://localhost:54000')
ro.hello
```

hello_client.rb

```
require 'drb/drb'

DRb.start_service
ro = DRbObject.new_with_uri('druby://localhost:54000')
ro.hello
```

約束事

- require 'drb/drb'
- DRb.start_service
- DRbObject.new_with_uri

元はこれ

```
class Hello
  def hello
    puts('Hello, World.')
  end
end

ro = Hello.new
ro.hello
```

```ruby
require 'drb/drb'

class Hello
  def hello
    puts('Hello, World.')
  end
end

DRb.start_service('druby://localhost:54000', Hello.new)
sleep
```

```ruby
require 'drb/drb'

DRb.start_service
ro = DRbObject.new_with_uri('druby://localhost:54000')
ro.hello
```

演習1

- hello_client.rbをなんども実行できますか？
- hello_server.rbを終了させてhello_client.rbを実行させるとどうなりますか？

使いどころ

- 空間
  - 一つのプロセスの扱える量を超える
- 時間
  - プロセスの寿命を超える (例 CGI)
- 機能による分割

演習2

- 共有されたメモ帳 (Hash) にいろいろ出し入れ／読み書きします
- 複数のプロセスがオブジェクトを交換する様子を感じてください

Hash

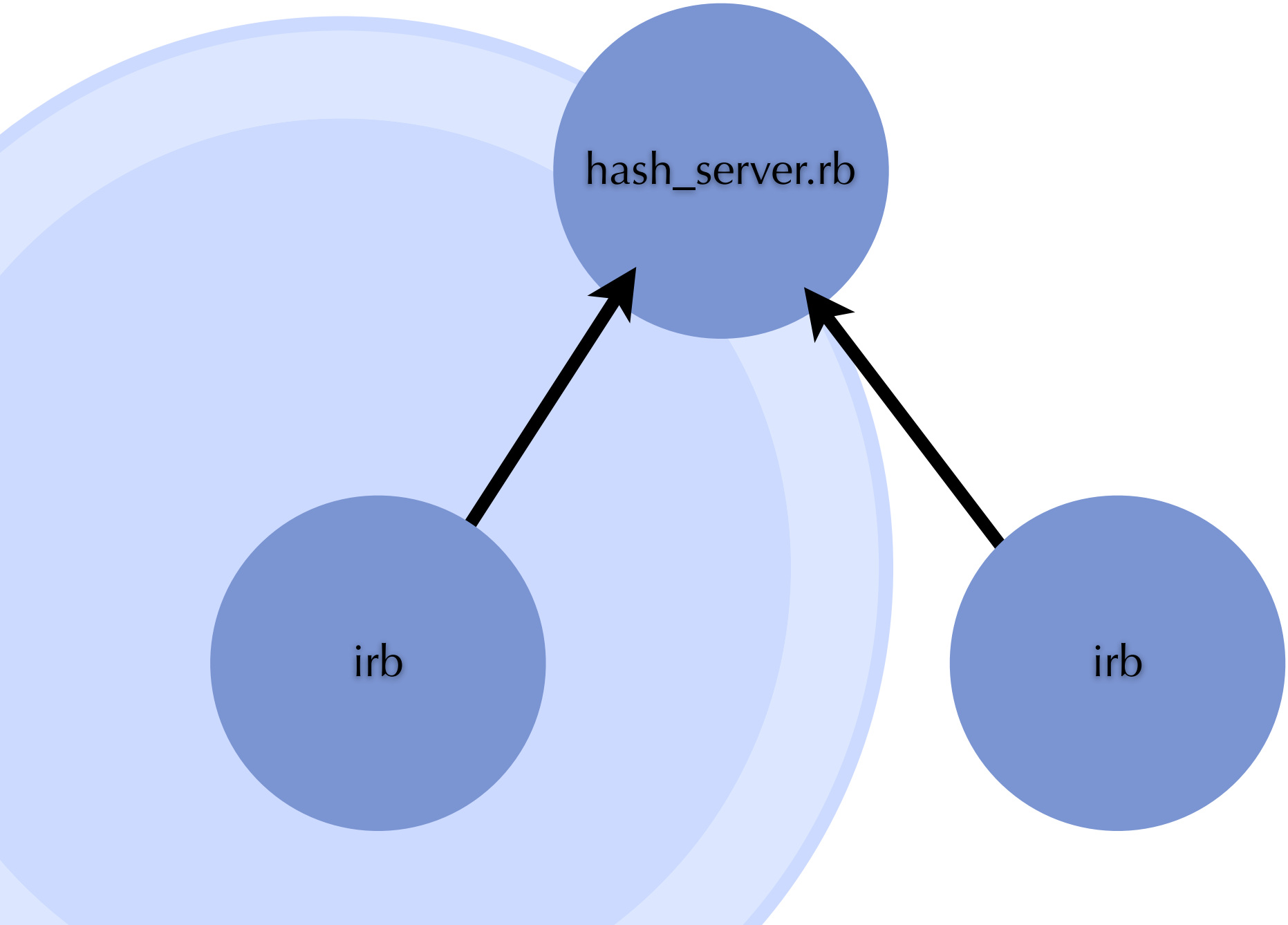
- 名前 (キー) と値を組で覚えておくもの
- 辞書に似てる
- Arrayとの違い
  - 並びの順序がない
  - キーは何でもよい (整数じゃなくても)

hash_server.rb

```
require 'drb/drb'
require 'pp'

front = Hash.new
DRb.start_service('druby://localhost:54300', front)
while true
  sleep 10
  pp front
end
```

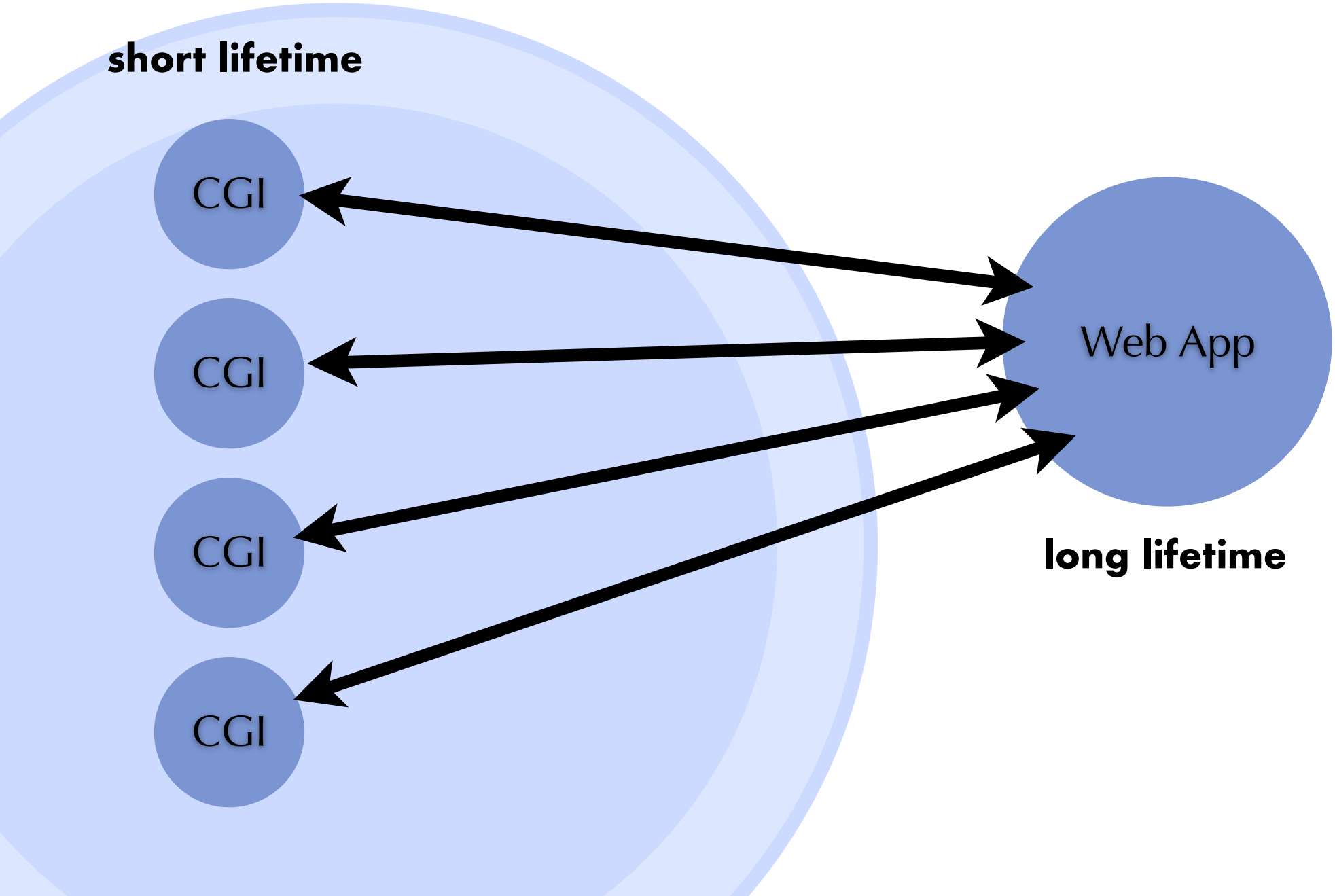
演習2



演習2

- どんなオブジェクトが入るか試して
  - "String"
  - 2012
  - Time.now
  - $stdout

CGIの例



WEBrickによるアプリ

- よく知られているスタイル
  - WEBrickがHTTPサーバでアプリももつ
- 実はCGIとしても使用できる

WEBrick::CGI

```
require 'webrick/cgi'

class MyCGI < WEBrick::CGI
  def do_GET(req, res)
    res["content-type"] = "text/plain"
    ret = "hoge\n"
    res.body = ret
  end
end

MyCGI.new.start()
```

start

```
WEBrick::CGI#start(env = ENV,
                   stdin = $stdin,
                   stdout = $stdout)
```

startメソッド

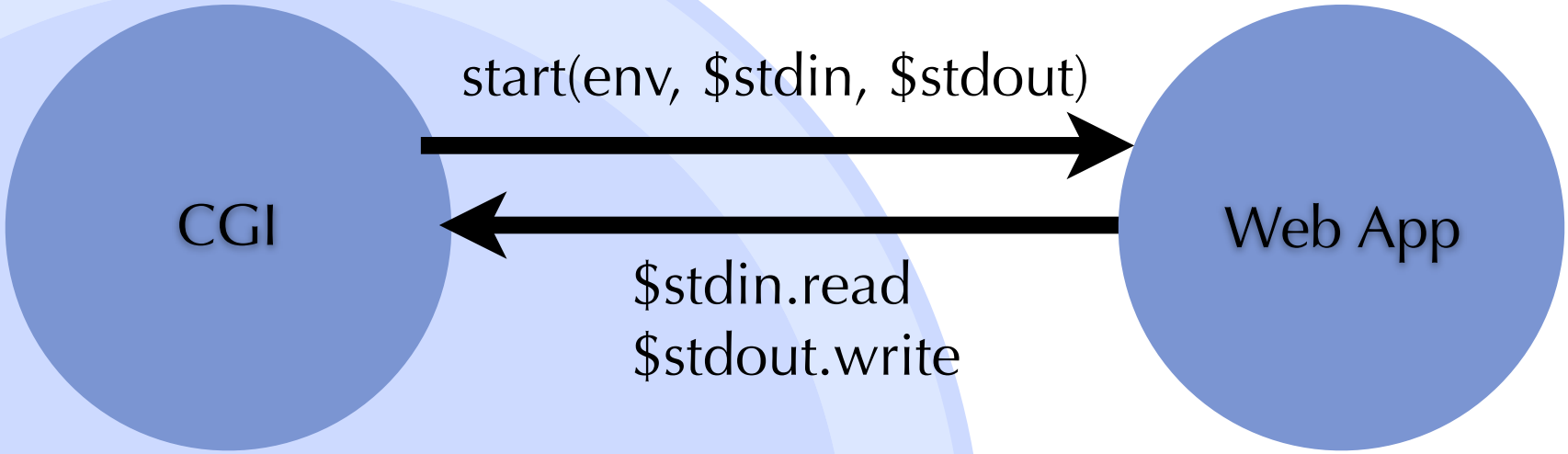
- 環境変数と$stdin, $stdoutを渡せば動く

m_seki's cgi

```
require 'drb/drb'

DRb.start_service('druby://localhost:0')
ro = DRbObject.new_with_uri('druby://localhost:50830')
ro.start(ENV.to_hash, $stdin, $stdout)
```

startのようす



るびま40号

Ruby コードの感想戦 【第 1 回】 WikiR

演習3

- 待ち行列を使った同期
- RubyのQueue
  - スレッド同期の仕組み

非同期と同期

- それぞれ勝手に動くプログラムを協調させる
- Queueは待合せと情報の交換を同時に行う

Queue

- FIFOバッファ
- pushとpop
- 空のときにpopするとブロックする
- データが届いたらまた動き出す

Queue

- スレッドの視点に立つと‥
  - 「待つ」のは取り出す係
  - 並べるのはオブジェクト
  - 並んで待つのではないことに注意

演習3

- データがない間、停まってるか？

```ruby
require 'drb/drb'
require 'thread'

DRb.start_service('druby://localhost:54320', Queue.new)
sleep
```

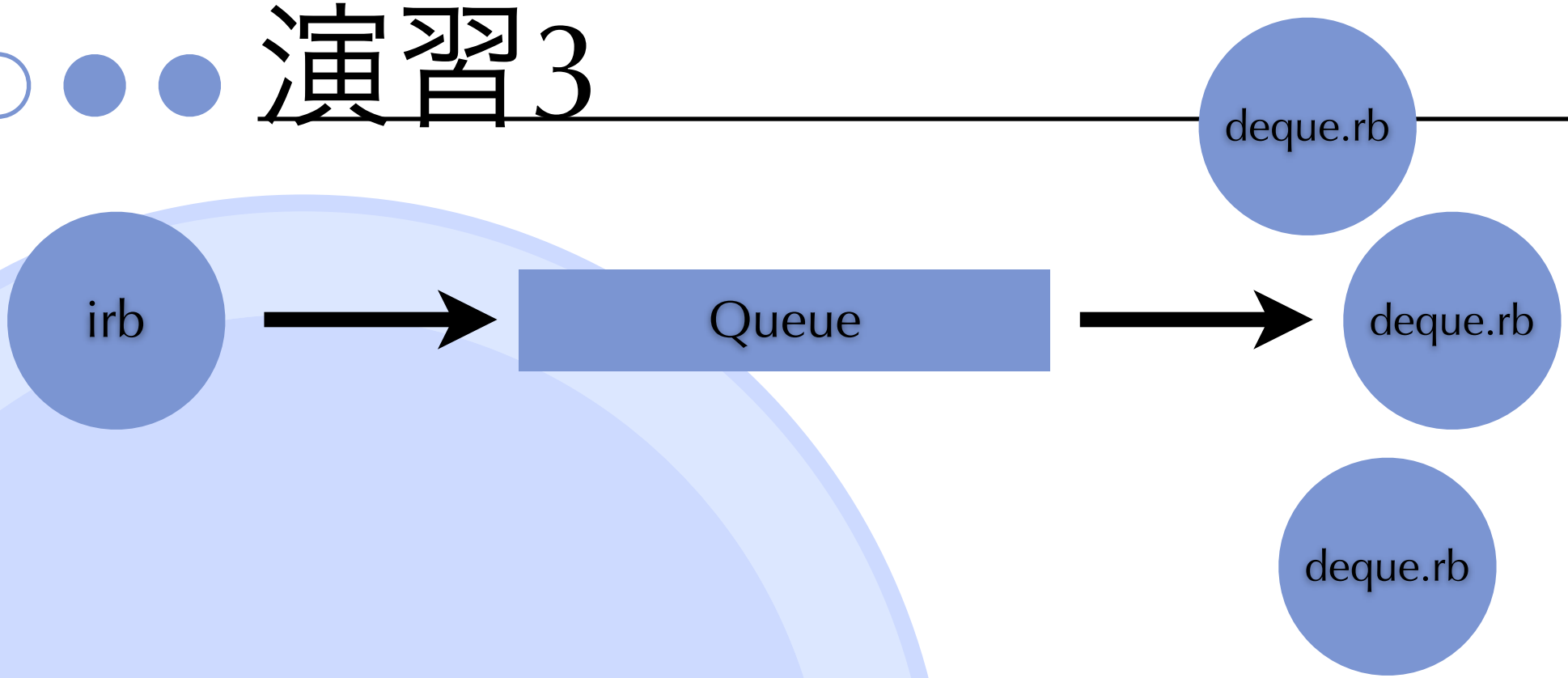
```ruby
require 'drb/drb

DRb.start_service
queue = DRbObject.
while true
  p queue.pop
  sleep(3)
end
```

Queueからデータを一つpopして印字する

ランダムに少しsleepする

演習3



- deque.rbを増やしたらどうなる?

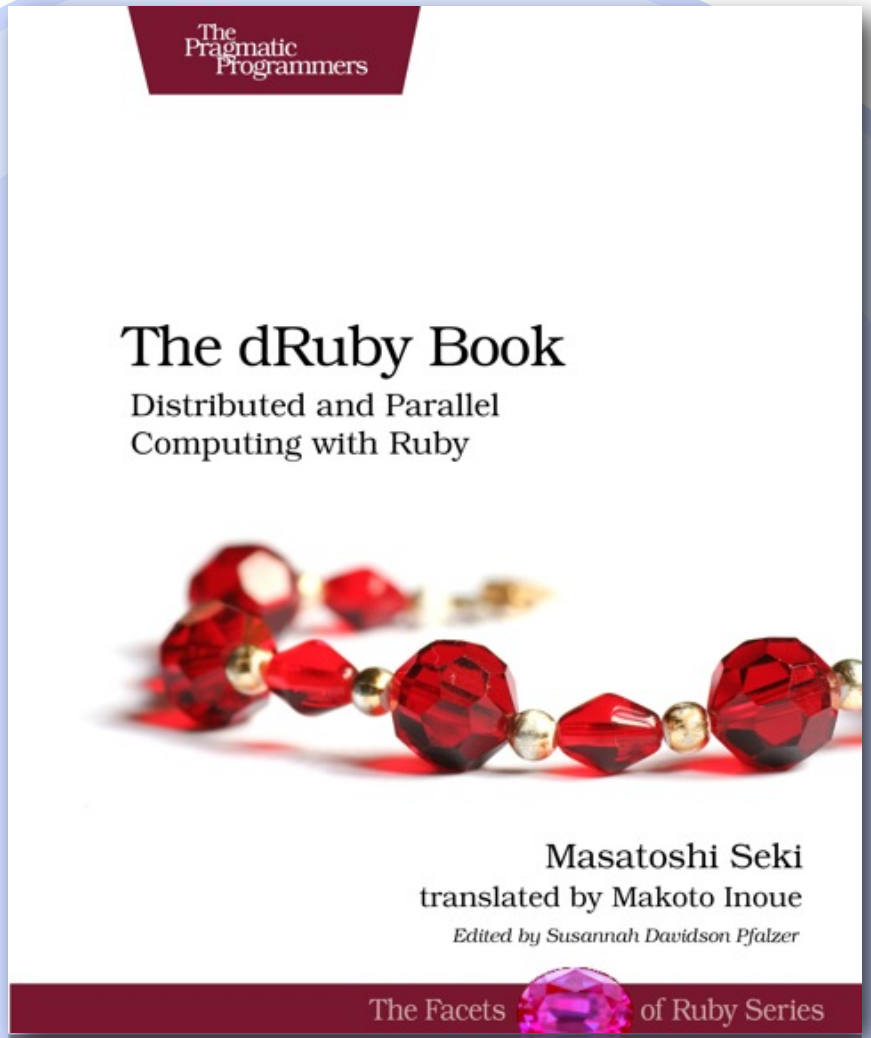
重要なことをもう一度

- もう一度

FAQ

- 英語版と日本語版のどちらを買うべきか？

両方です


The Pragmatic Programmers
The dRuby Book
Distributed and Parallel Computing with Ruby
Masatoshi Seki
translated by Makoto Inoue
Edited by Susannah Davidson Pfalzer
The Facets of Ruby Series


dRubyによる
分散・Webプログラミング
関 将俊 著
Ohmsha

まとめ

- dRubyの雰囲気を味わえた？

ふりかえり

演習4（やらない）

- Rindaを使った減らないQueue
- RDストリーム
  - http://www.druby.org/sidruby/6-3-basic-distributed-data-structures.html